Solving Linear Systems

(Numerical Recipes, Chap 2)

Jungdam Won

Computer Science & Engineering

Seoul National Univ.

Many contents are adopted from the slides of the Computer Animation course at SNU lectured by Jehee Lee

A System of Linear Equations

- Consider a matrix equation $\mathrm{A}\mathbf{x} = \mathbf{b}$

$$
\begin{gathered}
a_{00}x_0 + a_{01}x_1 + \cdots + a_{0n}x_n = b_0 \\
a_{10}x_0 + a_{11}x_1 + \cdots + a_{1n}x_n = b_1 \\
\vdots \\
a_{n0}x_0 + a_{n1}x_1 + \cdots + a_{nn}x_n = b_n
\end{gathered}
$$

- There are many different methods for solving this problem
  - We will not discuss how to solve it precisely rather we will discuss which method to choose for a given problem

What to Consider

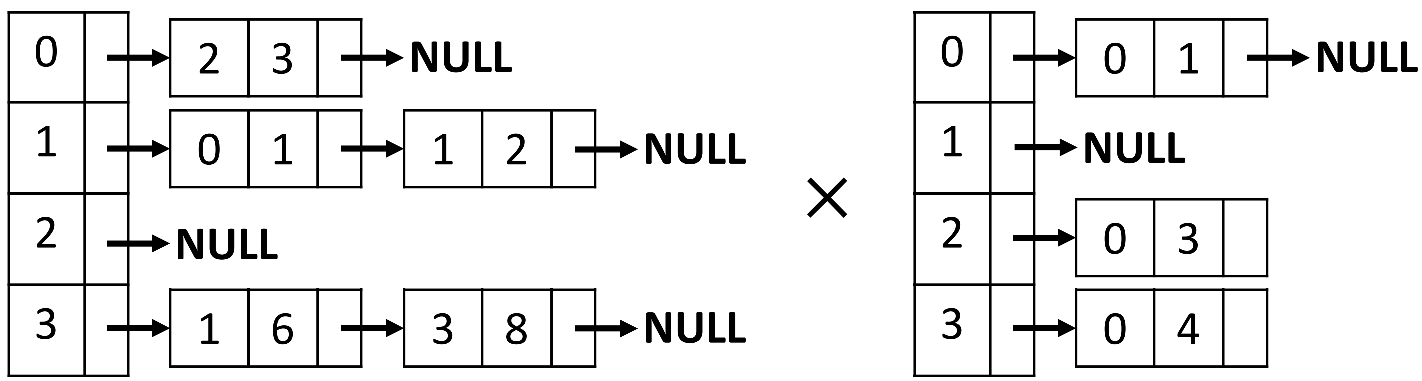
- Size
  - Consider a gray-scale image I of which size is (512x512 pixels) and a linear function $\mathbf{y} = F(\mathrm{I})$ where $\mathbf{y}$ represents a probability over 1K object classes
  - How much memory do we need to represent F?
    - F can be represented as a matrix of which size is 1000 x (512^2) assuming the input image flattened as a vector
    - 1000 x (512^2) x 8 (byte) ≈ 2GB
  - Computing the inverse of F requires additional 2GB to save the result

What to Consider

- Sparse vs. Dense
  - Many of linear systems have a matrix A in which almost all the elements are zeros
  - There exist special algorithms designated to sparse matrices for both the matrix representation and system solver

An Example of Sparse Matrix

$$\begin{bmatrix} 0 & 0 & 3 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 8 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 3 \\ 4 \end{bmatrix}$$



What to Consider

- Special properties
  - Symmetric $a_{ij} = a_{ji}$
  - Triangular $a_{ij} = 0 \text{ if } i < j$ or $a_{ij} = 0 \text{ if } i < j$
  - Banded $a_{ij} = 0 \;\text{ for }\; |i - j| > p \;\text{ where } p \text{ is bandwidth}$
  - Positive definite $\mathbf{x}^{\mathrm{T}} \mathrm{A} \mathbf{x} > 0 \quad \text{for all } \mathbf{x}, \qquad \mathbf{x} \neq 0$

What to Consider

- Singularity (i.e., $\det(A) = 0$)
  - Homogeneous systems
    - There exist non-zero solutions

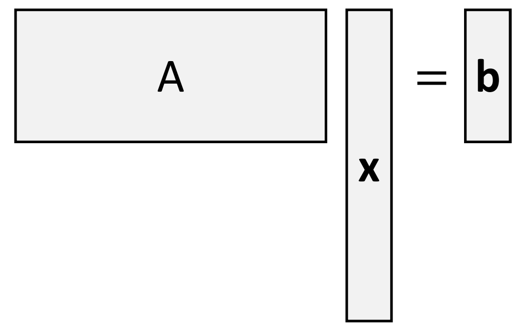
$$\mathrm{A}\mathbf{x} = 0$$

  - Non-homogeneous systems
    - Multiple solutions, or
    - No solution

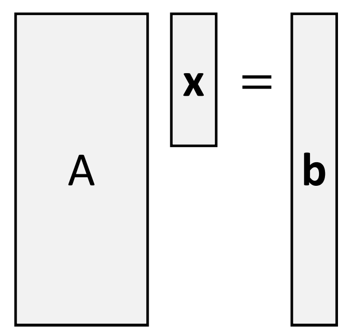
$$\mathrm{A}\mathbf{x} = \mathbf{b}$$

What to Consider

- Under-determined
  - Fewer equations (non redundant) than unknowns
    - A square singular matrix
    - # of rows < # of columns



- Over-determined
  - More equations than unknowns
    - # of rows > # of columns



Solution Methods

- Direct Methods
  - Guarantee to terminate within a finite number of steps
  - End with the exact solution
    - Provided that all arithmetic operations are exact
  - Ex) Gauss elimination
- Iterative Methods
  - Produce a sequence of approximations which hopefully converge to the solution
  - Commonly used with large sparse systems

Gauss Elimination

- Reduce a matrix A to a triangular matrix

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \Rightarrow \begin{bmatrix} a_{00}' & a_{01}' & a_{02}' & a_{03}' \\ 0 & a_{11}' & a_{12}' & a_{13}' \\ 0 & 0 & a_{22}' & a_{23}' \\ 0 & 0 & 0 & a_{33}' \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \end{bmatrix}$$

- Then, perform back substitution

Gauss Elimination

- Reduce a matrix A to a triangular matrix

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \Rightarrow \begin{bmatrix} a_{00}' & a_{01}' & a_{02}' & a_{03}' \\ 0 & a_{11}' & a_{12}' & a_{13}' \\ 0 & 0 & a_{22}' & a_{23}' \\ 0 & 0 & 0 & a_{33}' \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \end{bmatrix}$$

- Then, perform back substitution

$$x_3 = b_3'/a_{33}'$$

Gauss Elimination

- Reduce a matrix A to a triangular matrix

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \Rightarrow \begin{bmatrix} a_{00}' & a_{01}' & a_{02}' & a_{03}' \\ 0 & a_{11}' & a_{12}' & a_{13}' \\ 0 & 0 & a_{22}' & a_{23}' \\ 0 & 0 & 0 & a_{33}' \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \end{bmatrix}$$

- Then, perform back substitution

$$x_3 = b_3'/a_{33}'$$

$$x_2 = (b_2' - a_{23}'x_3)/a_{22}'$$

Gauss Elimination

- Reduce a matrix A to a triangular matrix

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \Rightarrow \begin{bmatrix} a_{00}' & a_{01}' & a_{02}' & a_{03}' \\ 0 & a_{11}' & a_{12}' & a_{13}' \\ 0 & 0 & a_{22}' & a_{23}' \\ 0 & 0 & 0 & a_{33}' \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \end{bmatrix}$$

- Then, perform back substitution

$$x_3 = b_3'/a_{33}'$$

$$x_2 = (b_2' - a_{23}'x_3)/a_{22}'$$

$$x_i = (b_i' - \sum_{j=i+1}^{N-1} a_{ij}'x_j)/a_{ii}'$$

Gauss Elimination

- Reduce a matrix A to a triangular matrix

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \Rightarrow \begin{bmatrix} a_{00}' & a_{01}' & a_{02}' & a_{03}' \\ 0 & a_{11}' & a_{12}' & a_{13}' \\ 0 & 0 & a_{22}' & a_{23}' \\ 0 & 0 & 0 & a_{33}' \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \end{bmatrix}$$

- Then, perform back substitution

- $O(N^3)$ computation for a $N \times N$ matrix

LU Decomposition

- Decompose a matrix A as a product of lower (L) and upper triangular (U) matrices

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$\mathrm{A}\mathbf{x} = (\mathrm{LU})\mathbf{x} = \mathrm{L}(\mathrm{U}\mathbf{x}) = \mathbf{b}$$

$$\Rightarrow \underbrace{\begin{bmatrix} \beta_{00} & 0 & 0 & 0 \\ \beta_{10} & \beta_{11} & 0 & 0 \\ \beta_{20} & \beta_{21} & \beta_{22} & 0 \\ \beta_{30} & \beta_{31} & \beta_{32} & \beta_{33} \end{bmatrix}}_{\mathrm{L}} \underbrace{\begin{bmatrix} \alpha_{00} & \alpha_{01} & \alpha_{02} & \alpha_{03} \\ 0 & \alpha_{11} & \alpha_{12} & \alpha_{13} \\ 0 & 0 & \alpha_{22} & \alpha_{23} \\ 0 & 0 & 0 & \alpha_{33} \end{bmatrix}}_{\mathrm{U}} \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \end{bmatrix}}_{\mathbf{b}}$$

LU Decomposition

- Decompose a matrix A as a product of lower (L) and upper triangular (U) matrices

$$\underbrace{\begin{bmatrix} \beta_{00} & 0 & 0 & 0 \\ \beta_{10} & \beta_{11} & 0 & 0 \\ \beta_{20} & \beta_{21} & \beta_{22} & 0 \\ \beta_{30} & \beta_{31} & \beta_{32} & \beta_{33} \end{bmatrix}}_{\mathrm{L}} \underbrace{\begin{bmatrix} \alpha_{00} & \alpha_{01} & \alpha_{02} & \alpha_{03} \\ 0 & \alpha_{11} & \alpha_{12} & \alpha_{13} \\ 0 & 0 & \alpha_{22} & \alpha_{23} \\ 0 & 0 & 0 & \alpha_{33} \end{bmatrix}}_{\mathrm{U}} \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \end{bmatrix}}_{\mathbf{b}}$$

- Solve $\mathrm{LU}\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$ to solve the system via forward and backward substitution
  - Let $\mathrm{U}\mathbf{x} = \mathbf{y}$
  - First solve $\mathrm{L}\mathbf{y} = \mathbf{b}$ for $\mathbf{y}$ then solve $\mathrm{U}\mathbf{x} = \mathbf{y}$ for $\mathbf{x}$

LU Decomposition

- If A is sparse, L and U may be sparse too whereas A^{-1} is likely to be dense in many cases

- LU decomposition is very efficient with triangular and band diagonal systems
  - LU decomposition and forward/backward substitution takes $\mathrm{O}(k^2N)$ operations, where k is the bandwidth

Matrix bandwidth

$$\begin{bmatrix} B_{11} & B_{12} & 0 & \cdots & \cdots & 0 \\ B_{21} & B_{22} & B_{23} & \ddots & \ddots & \vdots \\ 0 & B_{32} & B_{33} & B_{34} & \ddots & \vdots \\ \vdots & \ddots & B_{43} & B_{44} & B_{45} & 0 \\ \vdots & \ddots & \ddots & B_{54} & B_{55} & B_{56} \\ 0 & \cdots & \cdots & 0 & B_{65} & B_{66} \end{bmatrix}$$

https://en.wikipedia.org/wiki/Band_matrix

Cholesky Decomposition

- Suppose a matrix A is
  - Symmetric $\quad a_{ij} = a_{ji}$
  - Positive definite $\quad \mathbf{x}^{\mathrm{T}}\mathrm{A}\mathbf{x} > 0 \quad \text{for all } \mathbf{x}, \mathbf{x} \neq 0$

- Then, A can be decomposed as

$$\mathrm{L}\mathrm{L}^{\mathrm{T}} = \mathrm{A}$$

- Extremely stable numerically

QR Decomposition

- Decompose A such that $\mathrm{QR} = \mathrm{A}$
  - R is upper triangular
  - Q is orthogonal $\mathrm{QQ}^{\mathrm{T}} = \mathrm{I}$

- Generally, QR decomposition is slower than LU decomposition
- But, QR decomposition is useful for solving
  - The least squares problems $|A\mathbf{x} - \mathbf{b}|^2$ for overdetermined system

Jacobi Iteration

- Decompose a matrix A into three matrices

$$\mathrm{A}\mathbf{x} = (\mathrm{D} + \mathrm{L} + \mathrm{U})\mathbf{x} = \mathbf{b}$$

Upper triangular with zero diagonal
Lower triangular with zero diagonal
Diagonal matrix

- Then, run fixed-point iteration

$$\mathbf{x} = \mathrm{D}^{-1}(\mathbf{b} - \mathrm{L}\mathbf{x} - \mathrm{U}\mathbf{x}) \Longrightarrow \mathbf{x}^{(\mathrm{n}+1)} = \mathrm{D}^{-1}\left(\mathbf{b} - \mathrm{L}\mathbf{x}^{(\mathrm{n})} - \mathrm{U}\mathbf{x}^{(\mathrm{n})}\right)$$

- It converges if A is strictly diagonally dominant

$$|\mathrm{a}_{\mathrm{ii}}| > \sum_{j \neq i} |\mathrm{a}_{\mathrm{ij}}|$$

Gauss-Seidel Iteration

- The basic idea is the same but this makes use of "up-to-date" information

$$\mathbf{x} = \mathrm{D}^{-1}(\mathbf{b} - \mathrm{L}\mathbf{x} - \mathrm{U}\mathbf{x}) \Longrightarrow \boxed{\mathbf{x}^{(\mathrm{n}+1)} = \mathrm{D}^{-1}(\mathbf{b} - \mathrm{L}\mathbf{x}^{(\mathrm{n}+1)} - \mathrm{U}\mathbf{x}^{(\mathrm{n})})}$$

- Jacobi updates all-at-once while Gauss-Seidel updates row-by-row

$$\begin{aligned}
x_1^{(\mathrm{n}+1)} &= \phantom{0.2x_1^{(\mathrm{n}+1)}} + 0.2x_2^{(\mathrm{n})} + 0.2x_3^{(\mathrm{n})} \phantom{+ 0.2x_4^{(\mathrm{n})}} + 10 \\
x_2^{(\mathrm{n}+1)} &= 0.2x_1^{(\mathrm{n}+1)} + 0.2x_4^{(\mathrm{n})} + 20 \\
x_3^{(\mathrm{n}+1)} &= 0.2x_1^{(\mathrm{n}+1)} + 0.2x_4^{(\mathrm{n})} + 30 \\
x_4^{(\mathrm{n}+1)} &= +0.2x_2^{(\mathrm{n}+1)} + 0.2x_3^{(\mathrm{n}+1)} + 40
\end{aligned}$$

Gauss-Seidel Iteration

- The basic idea is the same but this makes use of "up-to-date" information

$$\mathbf{x} = \mathrm{D}^{-1}(\mathbf{b} - \mathrm{L}\mathbf{x} - \mathrm{U}\mathbf{x}) \Longrightarrow \boxed{\mathbf{x}^{(\mathrm{n}+1)} = \mathrm{D}^{-1}(\mathbf{b} - \mathrm{L}\mathbf{x}^{(\mathrm{n}+1)} - \mathrm{U}\mathbf{x}^{(\mathrm{n})})}$$

- It converges if A is strictly diagonally dominant
- Usually faster than Jacobi iteration
- There exist systems for which Jacobi converges, yet Gauss-Seidel doesn't

Singular Value Decomposition

- Decompose a matrix A with M rows and N columns $(\mathrm{M} \geq N)$

$$\mathrm{A} = \mathrm{UDV}^T$$

$$\begin{pmatrix} & & \\ & \mathrm{A} & \\ & & \end{pmatrix} = \underbrace{\begin{pmatrix} & & \\ & \mathrm{U} & \\ & & \end{pmatrix}}_{\text{Column Orthogonal}} \underbrace{\begin{pmatrix} w_1 & & \\ & \cdots & \\ & & w_{\mathrm{N}} \end{pmatrix}}_{\text{Diagonal}} \underbrace{\begin{pmatrix} & \mathrm{V} & \end{pmatrix}^{\mathrm{T}}}_{\text{Orthogonal}}$$

SVD for a Square Matrix

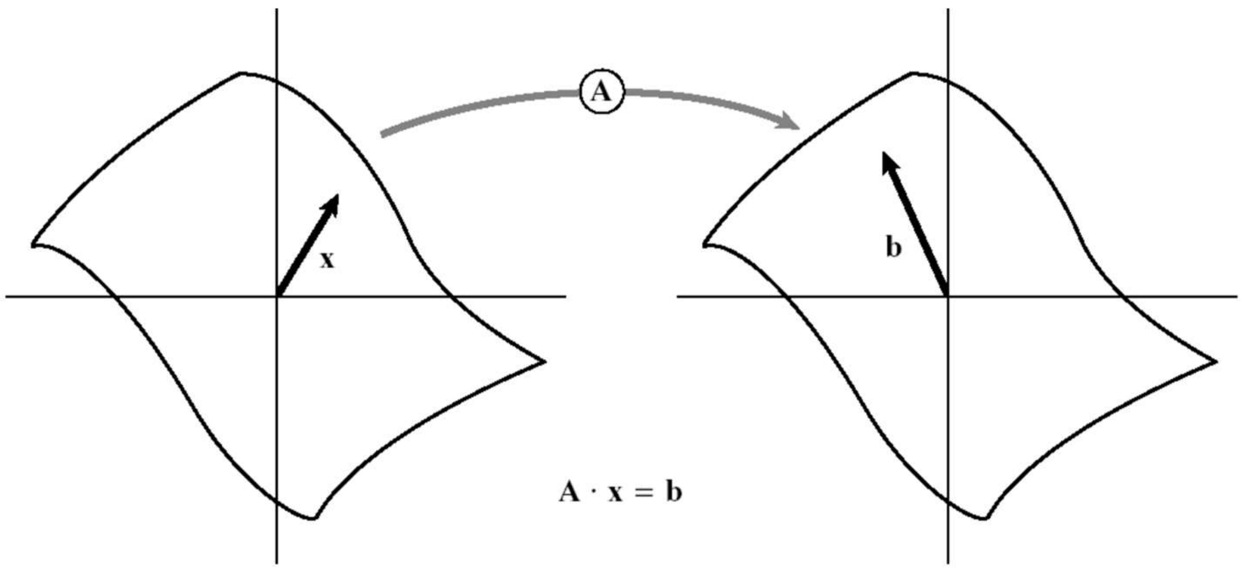
- If A is non-singular

$$\mathrm{A}^{-1} = \left(\mathrm{UDV}^{\mathrm{T}}\right)^{-1} = \mathrm{VD}^{-1}\mathrm{U}^{\mathrm{T}} = \mathrm{V}[\mathrm{diag}\left(\frac{1}{w_j}\right)]\mathrm{U}^{\mathrm{T}}$$

- If A is singular
  - Some of singular values will be zero

Linear Systems = Linear Transform

- Consider $\mathrm{A}\mathbf{x} = \mathbf{b}$ as a linear mapping from the vector space $\mathbf{x}$ to the vector space $\mathbf{b}$



Null space and Range

- Consider $\mathrm{A}\mathbf{x} = \mathbf{b}$ as a linear mapping from the vector space $\mathbf{x}$ to the vector space $\mathbf{b}$
  - The range of A is the subspace of $\mathbf{b}$ that can be **reached** by A
  - The null space of A is some subspace of $\mathbf{x}$ that is mapped to zero $\mathrm{A}\mathbf{x} = \mathbf{0}$
  - The rank of A equals to the dimension of the range of A
  - The nullity of A equals to the dimension of the null space of A

Rank + Nullity = N

Null space and Range

- The columns of U whose corresponding singular values are nonzero are orthonormal basis vectors of the range
- The columns of V whose corresponding singular values are zero are orthonormal basis vectors of the null space

$$\begin{pmatrix} & & \\ & \mathrm{A} & \\ & & \end{pmatrix} = \begin{pmatrix} & & \\ & \mathrm{U} & \\ & & \end{pmatrix} \begin{pmatrix} w_1 & & \\ & \cdots & \\ & & w_{\mathrm{N}} \end{pmatrix} \begin{pmatrix} & \mathrm{V} & \end{pmatrix}^{\mathrm{T}}$$

SVD for Underdetermined Problems

- If A is singular and $\mathbf{b}$ is in the range, the linear system has ***multiple solutions***

- We might want to pick the one with the smallest length $|\mathbf{x}|^2$

$$\mathbf{x} = \mathrm{V}[\mathrm{diag}\left(\frac{1}{w_j}\right)]\mathrm{U}^{\mathrm{T}}\mathbf{b}$$

Replace $\frac{1}{w_j}$ by zero if $w_j = 0$

SVD for Overdetermined Problems

- If A is singular and $\mathbf{b}$ is not in the range, the linear system has ***no solution***

- We can get the least squares solution that minimize the residual $|\mathrm{A}\mathbf{x} - \mathbf{b}|^2$

$$\mathbf{x} = \mathrm{V}[\mathrm{diag}\left(\frac{1}{w_j}\right)]\mathrm{U}^{\mathrm{T}}\mathbf{b}$$

Replace $\frac{1}{w_j}$ by zero if $w_j = 0$

SVD Summary

- SVD is very robust even when A is singular or near-singular
- Underdetermined and overdetermined systems can be handled uniformly
  - But, SVD is not a computationally efficient solution

$$O(M \cdot N \cdot \min(M, N))$$

$$\begin{pmatrix} \mathbf{x} \end{pmatrix} = \begin{pmatrix} \quad \mathbf{V} \quad \end{pmatrix} \cdot \begin{pmatrix} \mathrm{diag}(1/w_j) \end{pmatrix} \cdot \begin{pmatrix} \qquad \mathbf{U}^T \qquad \end{pmatrix} \cdot \begin{pmatrix} \\ \mathbf{b} \\ \\ \end{pmatrix}$$

(Moore-Penrose) Pseudo-Inverse

- Overdetermined systems $(m > n)$

$$\mathrm{A}\mathbf{x} = \mathbf{b}$$

$$\mathrm{A}^T\mathrm{A}\mathbf{x} = \mathrm{A}^T\mathbf{b}$$

$$(\mathrm{A}^T\mathrm{A})^{-1}\mathrm{A}^T\mathrm{A}\mathbf{x} = (\mathrm{A}^T\mathrm{A})^{-1}\mathrm{A}^T\mathbf{b}$$

$$\mathbf{x} = (\mathrm{A}^T\mathrm{A})^{-1}\mathrm{A}^T\mathbf{b}$$

$$\boxed{\mathrm{A}^\dagger = (\mathrm{A}^T\mathrm{A})^{-1}\mathrm{A}^T}$$

(Moore-Penrose) Pseudo-Inverse

- Underdetermined systems $(m < n)$

$$\mathrm{A}\mathbf{x} = \mathbf{b}$$

$$\underset{\mathbf{x}}{\operatorname{argmin}} |\mathrm{A}\mathbf{x} - \mathbf{b}|^2$$

$$\vdots$$

$$\mathbf{x} = \mathrm{A}^T (\mathrm{A}\mathrm{A}^T)^{-1} \mathbf{b}$$

$$\mathrm{A}^{\dagger} = \mathrm{A}^T (\mathrm{A}\mathrm{A}^T)^{-1} \mathbf{b}$$

A Damping Trick

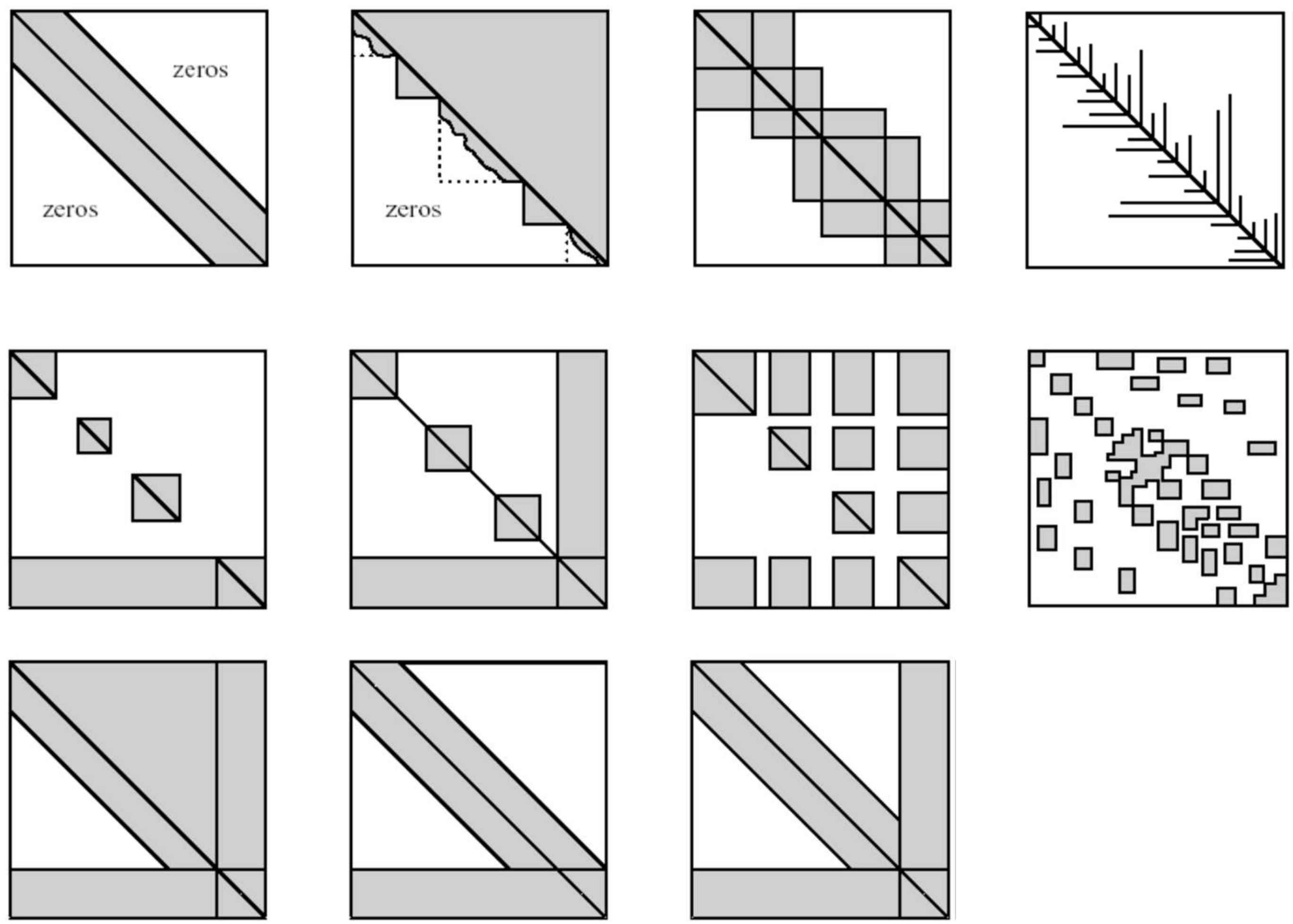
- Sometimes, if A is singular or near-singular, the behavior of many analytical methods could be numerically unstable

- With a bit of inaccuracy, we can make the behavior more stable by simply adding I to A

$$A \rightarrow A + \epsilon I$$

$$A^{-1} \rightarrow (A + \epsilon I)^{-1}$$

Sparse Linear Systems



Summary

- The structure of linear systems is well-understood
- If you can formulate your problem as a linear system, you are almost done. You can easily anticipate the performance and stability of the solution
- If your system matrices are sparse, then you are fortunate. It is very likely that the problem can be solved very efficiently